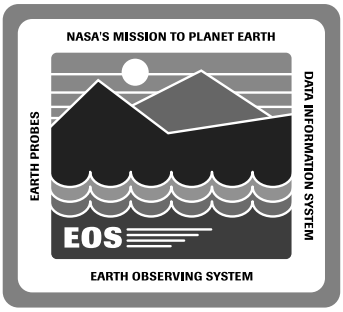
440-TP-008-001



Distributed and Parallel Processing For ECS Science Algorithms: A Trade Analysis

Technical Paper--Not intended for formal review or government approval.

February 1995

Prepared Under Contract NAS5-60000

RESPONSIBLE ENGINEER

Narayan S. Prasad 2/23/95

Narayan S. Prasad, PDPS Scientist/Engineer Date

EOSDIS Core System Project

SUBMITTED BY

Parag Ambardekar 2/23/95

Parag Ambardekar, PDPS Manager Date
EOSDIS Core System Project

Hughes Applied Information Systems
Landover, Maryland

This page intentionally left blank

Table of Contents

6. Evaluations

7. Recommendations and Conclusions

8. List of Acronyms

1.1 Trade description

The potential benefits from the areas of distributed and parallel computing for ECS science algorithms are investigated. If these technologies can be used to increase processing throughput, an indirect advantage will probably be to reduce the amount of disk staging space required. Distributed computing may also be useful for taking advantage of any available processing resources wherever they are located. All of these and other advantages must be evaluated against additional costs involved in writing distributed or parallel software and its potential for restriction of software portability. This is an ongoing trade scheduled to be completed at CDR. This trade analysis is intended to provide the most up-to-date information on processing technologies to scientists and software developers. It will also serve as reference for Data Processing System (DPS) hardware selection.

1.2 Organization

This paper is organized as follows:

An overall summary of the trade analysis is provided in Section 2 as Executive Summary. This section emphasizes the major conclusions arrived, and also recommends the appropriate choice of processing alternative for algorithms in general. The currently available processing alternatives are outlined in Section 3. Distributed computing with Open Software Foundation (OSF)/Distributed Computing Environment (DCE), parallel processing with Symmetric Multiprocessors (SMP), Distributed Memory Processors (DMP)/workstation cluster and Massively Parallel Processors (MPP) are explored. The ongoing prototyping efforts at the ECS Science and Technology Laboratory (STL) are summarized in Section 4. The selection criteria for each processing alternative and its applicability to processing scenario are discussed in Section 5. The pros and cons of each processing alternative and their evaluation based on prototyping results are detailed in Section 6. Section 7 draws conclusions and provides recommendations for the use of these technologies for ECS science processing.

1.3 Acknowledgments

The contributions of Marek Chmielowski and Scott Bramhall are sincerely appreciated.

1.4 Review and Approval

This trade study is not a formal paper. It has been approved as a working technical paper at the Office Manager level. Questions regarding technical information contained within this paper should be addressed to the following ECS and/or GSFC contacts:

- ECS Contacts

  Narayan Prasad, PDPS Scientist/Engineer

  (301)-925-0467

  nprasad@eos.hitc.com

- GSFC Contacts

  Steve Kempler, PDPS Manager

  (301)-286-7766

  steven.j.kempler@gsfc.nasa.gov

Questions concerning distribution or control of this document should be addressed to:

Data Management Office
The ECS Project Office
Hughes Applied Information Systems
1616 McCormick Dr.
Landover, MD 20785

1.5 Applicable and Reference Documents

1. Prototyping and Studies Plan for the ECS Project, 101-317-DV1-001, May 1993.
2. PDPS Prototyping at ECS Science and Technology Laboratory, Report #2, April 1994.
3. PDPS Prototyping at ECS Science and Technology Laboratory, Report #3, June 1994.
4. PDPS Prototyping at ECS Science and Technology Laboratory, Progress Report #4, 194-00569TPW, September 1994.
5. SDP Toolkit implementation with Pathfinder SSM/I Precipitation Rate algorithm, 194-430-TPW-001, November 1994.

2. Executive Summary

The availability of emerging processing technologies provide several choices for the implementation of ECS science algorithms. Distributed and parallel processing provide viable cost effective alternatives to conventional sequential processing. This trade, partly based on ECS Science and Technology Laboratory (STL) prototyping analyzes the applicability of using OSF/Distributed Computing Environment (DCE), SMP, DMP (including workstation cluster) and MPP for ECS science algorithms.

OSF/DCE provides an environment to harness the computing resource of several workstations or supercomputers. By providing services and tools, DCE supports the creation, use and maintenance of distributed applications in a heterogeneous distributed environment. It is based on a client/server model, Remote Procedure Call (RPC) mechanism and data sharing model. DCE is mainly applicable for coarse-grained applications, and may utilize resources geographically located. Development of DCE applications though difficult are partly aided by the availability of tools like Encompass/DCE from Open Environment Corporation. One major obstacle that DCE poses is that it assumes all libraries associated with the application be multithreaded. This can potentially be serious if the application contains heritage code that is unsuitable for a distributed environment. Encompass/DCE supposedly allows non-threadsafe libraries to be brought into a DCE environment. In short, DCE will be useful to science software that do not use Fortran (Fortran is not supported with DCE), exhibit coarse-grained parallelism, with parts of the application geographically distributed and on heterogeneous architectures (workstations, supercomputers, etc.).

Symmetric multiprocessing involves using a number of processors to increase aggregate throughput. All Central Processing Units (CPUs) are identical and share a common memory. Applications that exhibit any granularity of parallelism (coarse-, medium-, fine-grained) are suitable for SMP. It is ideal for parallel applications developers who do not wish to apply considerable effort to parallelize existing applications. A rich set of interactive and batch parallelization tools (native and third-party vendor) are available for symmetric multiprocessors. Many ECS science software will be benefited by this approach.

Distributed memory parallel processing using workstation clusters is newer and is gaining popularity because it is economical. Because the memory is distributed, unlike shared memory, the programmer must explicitly move data to the distributed CPUs for processing. In this case Input/Output (I/O) remain a serious bottleneck, especially if an science software is I/O intensive. Applications that exhibit any granularity of parallelism (coarse-, medium-, fine-grained) are suitable for DMP workstation cluster. Programming distributed applications is considerably more difficult. However, a rich set of parallelization tools are available that automatically generate code for communication among distributed processors. The speed of interconnect, and the speed of the slowest machine on the cluster are factors to be considered. Workstation cluster can be used as a development environment for MPP architectures. With advanced high performance interconnects available, and more work targeted toward I/O performance in a cluster (parallel I/O),

workstation cluster technology is certainly poised to take on the challenges that once used to be exclusively dominated by MPPs.

Massively parallel processing involves a very large number of processors. They follow the distributed memory model, therefore, requiring explicit communication among processors. The main advantage is that MPPs provide multiple I/O paths into the system. If the data can be partitioned among different I/O channels, then current MPPs can provide acceptable solution. The parallelization tools discussed earlier for DMP workstation cluster are also available for select MPP machines. This allows parallel applications to be first developed on workstation cluster before migrating to MPP architectures. Applications that are heavily I/O and CPU intensive, and exhibit a high degree of parallelism should consider MPPs. MPPs could potentially benefit ECS "tall pole" algorithms.

3. Major Trade Alternatives

3.1 Introduction

There are several processing alternatives available for the implementation of ECS science algorithms. The following sections discuss the currently available alternatives.

3.1.1 Distributed computing with DCE

One such environment that can harness the computing resource of several workstations or mainframe supercomputers is the OSF/Distributed Computing Environment (DCE). DCE provides services and tools that support the creation, use and maintenance of distributed applications in a heterogeneous computing environment. The machines can physically be located anywhere, and are connected over the network. DCE provides interoperability and portability across heterogeneous platforms. DCE is based on three distributed computing models - Client/Server, Remote Procedure Call (RPC) and Data Sharing. The client/server model is a way of organizing a distributed application. The distributed application is divided into two parts, one part residing on each of the two computers that will be communicating during the distributed computation. The RPC model is a way of communicating between parts of a distributed application. In this model, the client makes a procedure call, which is translated into network communications by the underlying RPC mechanism. The server receives a request, executes the procedure, returning the results to the client. The data sharing model is a way of handling data in a distributed system. In this model, the data is shared by distributing it throughout the system. In data sharing, a copy of the server's data is sent to the client, and the client accesses the file locally.

3.1.2 Parallel processing with SMP and DMP

The fundamental idea behind parallel computing is to task CPUs as a team to solve single or multiple problems. Parallel processing has historically been hardware- and performance-driven. Parallel systems are generally classified by the method in which they process data and instructions (*i.e.* in singular or multiple data or instruction streams). There are four combinations of ways of processing instructions and data:

- SISD (Single-Instruction, Single-Data)
- SIMD (Single-Instruction, Multiple-Data)
- MISD (Multiple-Instruction, Single-Data)
- MIMD (Multiple-Instruction, Multiple-Data)

The two most common architectures are SIMD and MIMD. In SIMD, each processor executes exactly the same command at each clock cycle on multiple data. MIMD on the other hand allows multiple instructions on multiple data during each clock cycle. Furthermore, depending on the memory organization for parallel systems, they can either share memory or have distributed memories as discussed later.

3.1.2.1 Symmetric multiprocessing

Symmetric multiprocessing involves using a number of processors (probably not a "massive" number) to increase aggregate throughput of a computer system. All CPUs are identical and any CPU can execute both user code and kernel (operating system) code. The CPUs operate on a peer basis, executing a single copy of the operating system executive or kernel. There is no designated master CPU except during system startup and diagnostic operations. Any process (program) in any state can execute on any CPU. The application developer can simply recompile existing codes and let the compiler detect whatever parallelization it is capable of detecting. This is called shared memory (SM) parallel. SM typically results in fairly fine-grained parallelization (for example, at the loop level). This design is called tightly-coupled, because the processors must coordinate access to the common memory. Each processor has cache memory which allows it to limit access to the common memory and, thereby, minimize its use of the system bus. Parallel computation can occur at the statement, procedure, or program level, so these systems can accommodate anything from fine- to coarse-grained parallelism. The SGI Challenge, SGI Power Challenge, SGI Power Onyx, Convex Exemplar, Cray Superserver 6400, DEC TURBOlaser, etc. are a few examples of SMP platforms.

3.1.2.2 Distributed memory parallel processing

The support of programs that explicitly support parallelization through message passing (MP) is also widely used. Parallel Virtual Machine (PVM) from Oak Ridge National Laboratories is one of the MP libraries that is popular today. The benefits of using the MP programming model through PVM (or any other MP libraries) is that the application will often execute on a variety of platforms. This type of design is described as loosely-coupled, meaning that each processor is almost entirely self-sufficient due to its local memory. Distributed-memory multiprocessors are most useful for problems that can be broken down into totally independent or unrelated parts, each of which requires extensive computation or other activity (like I/O, etc.). The Cray J916 is an example of a small size DMP and also a vector computer. It should be noted that an SMP or an SMP cluster can also be used as a distributed memory parallel processor.

3.1.2.3 Distributed/parallel computing on clustered platforms

Distributed/parallel computing has become increasingly popular recently. Distributing parallel applications across clustered platforms fall under the MP paradigm. The message passing method can run parallel applications across clustered heterogeneous platforms interconnected by a high performance network. In addition to PVM which add parallel support for computers distributed across a network, commercial packages like Linda (Scientific Computing, Inc.) and Express (Parasoft, Inc.) also add parallel support for multiple processors. Message Passing Interface (MPI) from Oak Ridge National Laboratories, a standard message passing interface for distributed memory machines is widely gaining acceptance. Digital's Alpha AXP farms provide comprehensive solutions with heterogeneous workstation clusters through software for compute sharing, file systems support, and parallel computing. Their LSF software provide clustering capability by integrating UNIX systems and low-cost servers to effectively use the computing power

available from distributed systems. The components of the cluster, Alpha AXP systems (or any supported UNIX platform), Ethernet, FDDI, or GIGAswitch, UNIX cluster software LSF, PVM and PolyCenter can together increase throughput (through batch queuing, load leveling, interactive job management), perform parallel processing and utilize unused distributed compute cycles.

3.1.3 Parallel processing using Massively Parallel Processors

Although the term massively parallel processing is overloaded, it typically refers to processing done on numerous processors which may or may not be small. The paradigms discussed previously for Parallel Processing apply to massively parallel processing using MPPs. Table 1 lists a few MPPs along with the number of processors.

Table 1. Overview of Current MPPs

Make	Technology	Number of Processors
Maspar MP-1	SIMD	16384
TMC CM-2	SIMD	2048
TMC CM-5	MIMD	16384
nCUBE2	MIMD	1024
nCUBE3	MIMD	65536
iPSC/860	MIMD	128
Intel Delta	MIMD	512
Intel Paragon	MIMD	4096
IBM SP-2	MIMD	256
Cray T3D	MIMD	32 to 2048

4. Analysis Approach

4.1 Introduction

The successful operation of the science software in the DPS is crucial to the success of the mission as a whole. The design and the operational characteristics of the DPS will determine the interfaces between the science software and the systems. Both heritage code (e.g. CERES/ERBE) and new code developed must be integrated into the DPS environment, how distributed science software will exercise the capacity of the planning and data processing system needs to be understood, emerging technologies must be identified, the applicability of emerging technologies for ECS science processing must be assessed, and the risks associated with new technologies must be evaluated.

To understand and evaluate concepts for processing, under the Prototyping and Studies Plan for the ECS Project [1], a distributed/parallel test bed consisting of several workstations and server class machines connected via Fiber Distributed Data Interface (FDDI) has been established at the ECS STL. The test bed provides the hardware and software infrastructure to support initial parallel processing investigations within Hughes. Besides the test bed, vendor loaned hardware and hardware at select High Performance Computing Center (HPCC) sites are also used. The goals of the distributed and parallel processing trade analysis task are to:

- Analyze representative ECS science algorithms to understand applicability to distributed and parallel computing.
- Understand portability issues when migrating from existing serial science software to distributed or parallel environments.
- Evaluate the status of automatic parallelization compilers that would benefit the science community at large to aid in the parallelization of applications.
- Identify issues to facilitate algorithm integration and testing using the Science Data Processing (SDP) Toolkit
- Provide information for DPS hardware requirements and architecture selection.
- Develop an ECS high performance computing benchmark suite.
- Understand parallel processing languages like Fortran 90, High Performance Fortran (HPF), etc. This work is tentatively planned for post PDR timeframe.
- Provide science software experiences and lessons learned with these emerging technologies to the science community in general.

4.2 Prototyping science software

The Pathfinder (Advanced Very High Resolution Radiometer (AVHRR)/Land Global Area Coverage (GAC) and Special Sensor Microwave/Imager (SSM/I) Precipitation Rate) and SeaWinds algorithms have been characterized and analyzed. A strategy was then

developed for each algorithm and applied to various processing alternatives. The SDP Toolkit was also embedded in the science software to study performance. The results of these activities have been reported from time to time to ESDIS and instrument teams as presentations, lab demonstrations and as informal reports published through ECS Data Handling System (EDHS) [2], [3], [4], and [5].

The algorithms chosen may not be representative in terms of ECS data volumes and (Millions of Floating Point Operations) MFLOPs. They, however, provide insight into the kind of processing that is expected for ECS. Also, by analyzing these algorithms, it is possible to derive some general results on processing that will be of use to the science community at large.

5. Selection Criteria

5.1 Introduction

Whether a new parallel application is designed or an existing application from a serial system is being ported, the key to success is being able to see the application from a new viewpoint and disregarding older, more traditional solutions. Parallelism should not be forced on the problem. Instead, the point is to find and exploit any potential parallelism inherent in the problem. In some applications, the calculations on one set of data are completely independent of calculations on other sets of data (*i.e.*, there is no communication between the calculations), and the calculations all require approximately the same amount of time. While in other applications, the time required may not be the same. It is safe to generalize that most ECS algorithms have inherently parallelism built into it. The instruments that collect data are either scanners, imagers, etc. that repeatedly collect data per orbit pass or swath. The data are then subjected to orbit-by-orbit, pixel-by-pixel processing (there may be exceptions with ECS Level 4 algorithms). The following sections identify criteria that would allow a science application developer to carefully match the requirements of their application with a processing alternative.

5.2 Selection criteria for using DCE

A cluster of workstations with DCE can be used as a distributed cooperative computing system. The key to success is to find concurrency and parallelism in an algorithm. The following identify situations when DCE can potentially be useful:

- When an application exhibits coarse-grained parallelism, for example at the orbit level, each orbit can be processed independently on multiple processors in a client-server mode. This is accomplished by placing the complete sequential science software on each processor and segmenting the data set so that each processor handles one segment at a time. The DCE threads service is used to perform individual tasks, which are then mapped to remote processors using the RPC mechanisms. DCE threads provide portable facilities that support concurrent programming, allowing an application to perform many actions simultaneously. The threads service includes operations to create and control multiple threads of execution in a single process, and to synchronize access to global data within an application. Because a server process using threads can handle many clients at the same time, the threads service is ideally suited to handle multiple clients in client-server based applications. The lesser the dependency among the segments, the lower the overhead due to communication, and better the performance.

- For applications where I/O operations which are typically slower and can be localized. In such cases, the computational operations can be performed independently, usually targeting platforms that are more suitable for such an operation. Again, DCE threads service and RPC mechanisms can be used.

- When parts of an application are required to be distributed geographically. For such cases, the full services of DCE's high level calling functions can be used to create distributed applications.

5.3 Selection criteria for parallel processing with SMP

Parallel processing on SMP is potentially suitable for algorithms that satisfy the following criteria:

- When processing is based on orbits, scenes, cells, frequencies, etc. and calculations performed on one portion of the data are independent of other portions. Such an algorithm is said to exhibit data parallelism in a coarse-grained form.
- When applications exhibit any granularity of parallelism (coarse-, medium-, fine-grained). The architecture maintains a consistent view of shared memory for all processors. Communication among processors is implicit in the shared memory model.
- When reasonably good performance gain through parallelization needs to be achieved with minimal effort.
- When scalability to a large number of processors is not a concern. Currently, a single SMP (some models) can have up to 32 processors.

5.4 Selection criteria for workstation cluster

DMP workstation cluster can potentially be used when:

- cost is a major concern.
- unused CPU cycles from several individual workstations are available. The cumulative computing power and affordable memory of the current generation of workstations make it possible to run many of the tasks currently running on supercomputers.
- algorithms inherently show any granularity of parallelism.
- data volumes are small and CPU workload is substantially greater than I/O workload. PDPS prototyping has indicated that when serial I/O is used and data volume is large, substantial amount of time is spent in moving data back and forth the distributed CPUs. With parallel I/O where data are arranged in such a way that individual processors can read data asynchronously, the data movement among processors was reduced drastically and performance improved tremendously. Parallel I/O for workstation cluster is currently a hot topic in scientific research.

5.5 Selection criteria for MPP

- When the algorithm lends itself to parallelism at a very fine-grained level to take advantage of the numerous processors available.
- When extremely high bandwidths and very low latencies are required by the application for interconnects among the processors.

- When extremely high I/O bandwidth is necessary.
- When homogeneity of processors is essential to the application.
- When data volumes and CPU workload are extremely large (ECS tall pole algorithms).
- When the application is expected to grow in terms of processing requirements, therefore scalability of processors is important.
- When the application requires large quantities of memory.

6. Evaluations

6.1 Introduction

As part of the PDPS prototyping in the ECS STL, the Pathfinder AVHRR/Land algorithm from NASA/GSFC was used to determine if DCE is a viable alternative for ECS science processing. Detailed analysis and results can be found in [3]. The following sections are based on experiences and lessons learned with PDPS prototypes.

6.1.1 Advantages of using DCE

- Distributed clustered systems are economical, and the user can decide the configuration depending upon the requirements. Configuration can also be changed with changing requirements.
- Allows sequential algorithms to be used in a coarse-grained distributed mode.
- Any combination of workstations, supercomputers and even multiprocessor systems situated anywhere on the network can be used for distributing an application.
- Allows for targeting machines for I/O and CPU intensive portions of the program
- A DCE cluster provides several advantages over a single system. Expensive storage resources are shared by the nodes in the cluster. It provides cost effective high availability by providing features that increase the time during which an application is available. Layered database products specifically designed for a shared disk environment perform better when run on a DCE cluster.
- The DCE Threads Service provides a simple programming model for building concurrent applications.
- A DCE cluster provides transparent multiprocessing support where applications need not know whether threads are executing on one or several processors.
- Multiple DCE threads share a common address space and resources, and can be mapped to available processes.
- Multiple threads can reduce the complexity of some applications that are inherently suited for threads.
- DCE stubs perform data conversion in a heterogeneous environment making it interoperable.
- Object Oriented (OO) DCE is now available from most vendors for science algorithms designed using OO methodologies.

6.1.2 Disadvantages with DCE

- The performance depends on the network that cluster the workstations together. Relatively low bandwidth of the interconnect, and the high communication overhead severely limit the types of algorithms that can effectively be processed in this mode. Science software that work well are those in which there is minimal coupling required between cooperating processors.
- Managing multiple threads can become complex in an application.
- Fortran is not supported by DCE.
- DCE threads are not suitable for fine-grained parallelism (at the loop level).
- DCE does not support non-threadsafe libraries (libraries that are not designed for concurrent or parallel operations).

6.13 Performance of DCE applications

DCE threads improve the performance (throughput, computational speed, responsiveness, or some combination of these) of a program. Multiple threads are useful in a multiprocessor system where threads run concurrently on separate processors. In addition, multiple threads also improve program performance on single processors systems by permitting the overlap of input and output or other slower operations with compute intensive operations. Also, by distributing parts of the application, the overall throughput can be increased.

6.1.4 Portability of DCE applications

Portability across the entire distributed environment depends upon how portable the science software is. Besides, portability across the distributed environment also requires file system and I/O transparency. The DCE file system gives users a uniform name space, file location transparency, and high availability. This means, data available in one location can be shared by processors in another location.

6.1.5 DCE Issues

- SDP Toolkit requires multithreaded support in order to be used in a DCE environment
- Hierarchical Data Format (HDF)-EOS (HDF with EOS extensions) libraries requires multithreaded support to be used in a DCE environment
- Encompass/DCE from Open Environment Corporation can automate DCE threads, RPC, Directory and Security Services, and provide simulation of multithreaded servers for applications written in languages or using libraries that do not support multithreading.

6.2 Evaluation of SMP

As part of the PDPS prototyping in the ECS STL, the Pathfinder SSM/I Precipitation Rate algorithm from NASA/MSFC was used to determine if SMP is a viable alternative for ECS science processing in terms of price/performance, ease of creating new applications, ease of converting existing serial applications, maturity of parallelization compilers, etc. A symmetric multiprocessor SGI Challenge XL loaned by Silicon Graphics, Inc. was used for the evaluation. The algorithm was characterized and a processing strategy developed. This strategy was then implemented using automatic and interactive parallelization tools. The applicability of SMP was demonstrated to ESDIS and instrument teams. Detailed analysis and results can be found in [5].

6.2.1 Parallelization tools for SMP

The SGI Challenge XL has the Power Fortran Accelerator (PFA) and Power C Analyzer (PCA). The PFA is a native Fortran 77 source-to-source preprocessor that enables existing Fortran 77 programs to run efficiently on SGI Power Series multiprocessor systems. PFA analyzes a program and identifies loops that do not contain data dependencies. Such loops are safe to execute in parallel (concurrently). PFA automatically inserts special compiler directives in a modified copy of the original source code. PFA also produces a number of files containing code and other information to run the program simultaneously on multiple processors. Because the directives inserted by PFA look like standard Fortran 77 comment statements, PFA does not affect the portability of the code to non-SGI systems. The PFA inserts code in a modified copy of the original source code. The listing file optionally generated by PFA can be used to identify potential data dependencies that prevented PFA from running a loop in parallel. The object files prepared by PFA are fully compatible with object files prepared for a serial compiler. They can be freely combined for execution. Serial C programs can be parallelized on SGI multiprocessor systems using the PCA. It is a C code parallelizing compiler that automatically analyzes sequential code to determine where loops can run in parallel, and then generates object code that can use multiple processors.

It should be emphasized that only the SGI Challenge was evaluated at the STL. There are similar native parallelization preprocessors on other SMP platforms. Other third-party parallelization tools are also available for SMP architectures. One such tool that is very advanced is Forge 90/xHPF from Applied Parallel Research (APR), Inc. Forge 90 is a GUI-based interactive parallelization preprocessor. xHPF is a batch preprocessor. Both Forge 90 and xHPF automatically, or by programmer choice parallelize existing serial programs for shared memory multiprocessor systems.

6.2.2 Advantages of SMP

- The shared memory model in SMP makes program for multiprocessing easy.
- SMPs are good for programmers who like reasonable performance albeit with minimal effort.

- With native and third-party parallelization compilers, achieving high performance is relatively painless.
- Parallelization tools shift a fair amount of responsibility away from the programmer.
- The SMP parallelization compilers retain source code portability. The parallelization directives and code modifications by the tool are performed on a copy of the original source code.
- SMP shares the same memory subsystem, let alone the same cabinet.
- By sharing common memory and I/O subsystems, the processor may share peripherals, eliminating redundancy and wasted disk space.
- The operating system automatically ensures that all CPUs are kept busy as long as there are executable processes available. However, the programmer is still responsible for balancing the work on individual processors to maximize CPU utilization.
- Only one executable of the application needs to be prepared because all the processors are identical.

6.2.3 Disadvantages of SMP

- SMP is not very scalable. The number of slots limit the addition of processors.
- The bus architecture could be a bottleneck for very I/O intensive multiprocessor applications, although SMP buses are high speed.

6.2.4 SMP Issues

- The shared memory model allows non-threadsafe libraries to be used in serial (HDF, SDP Toolkit, etc.) by switching between serial and parallel modes [5].
- Switching between serial and parallel modes can affect overall performance of a parallel science software, and may not work for all cases. Some algorithms may require the use of SDP Toolkit functions within a parallel region (e.g. geolocating pixels/cells when processing pixels/cells in parallel) [5].
- SDP Toolkit would require special handling of global address space in shared memory model [5].
- HDF-EOS libraries can be used only in serial mode on most machines. Support for parallel I/O is available only on select machines [5]. However, lack of industry-wide standards seriously limit the use of parallel I/O, especially for enhancing the SDP Toolkit and HDF libraries for a parallel environment.

6.2.5 Performance

In SMP systems as the processing load increases, several CPUs are available to service the increasing number of processes. Each process is given more time with a CPU, each process spends less time waiting, and the system spends less time switching from one process to the next. Performance is, thereby, greatly increased. Programs can be

automatically or manually tailored to use multiple CPUs simultaneously, and thus obtain a dramatic increase in execution speed. Parallel applications can be structured to adapt themselves to the number of CPUs present in the system. Thus, the number of CPUs is simply a tuning parameter: the same parallel application can be run on systems of different configurations without any major modifications to the existing software.

6.3 Evaluation of DMP using workstation cluster

Again, as part of the PDPS prototyping in the ECS STL, the Pathfinder SSM/I Precipitation Rate algorithm from NASA/MSFC was used to determine if DMP/workstation cluster is a viable alternative for ECS science processing in terms of price/performance, ease of creating new applications, ease of converting existing serial applications, maturity of parallelization compilers, etc. The ECS distributed/parallel test bed at the STL was used for the evaluation. The algorithm was characterized and a parallelization strategy developed. This strategy was then implemented using automatic and interactive parallelization tools. A lab demonstration summarizing the experiences and lessons learned was presented to ESDIS and instrument teams.

6.3.1 Parallelization tools for DMP using workstation cluster

APR's parallelizing compilers like Forge 90/xHPF generate a parallelized SPMD (Single Program, Multiple Data) Fortran 77 program for execution on a distributed memory multiprocessor system. Forge's distributed memory parallelizer spreads loops and distributes arrays across multiple processors of a distributed memory machine or a cluster of workstations. The automatic parallelization mode automatically identifies which loops to parallelize. Forge 90 also allows interactive selection of loops. Data communication calls to the chosen message passing library (PVM, Express, Linda, etc. are supported) and other APR runtime libraries are inserted automatically around and within distributed loops to move the data as it is needed. The parallelization tool inserts code in a modified copy of the original source code.

Clustering shared-memory supercomputing servers has also been demonstrated by leading vendors like SGI, Convex, etc. They are viable alternatives to the more expensive MPP architectures (to be discussed later). APR has recently announced a parallelization tool that allows the combination of both shared memory and distributed memory strategies to address multiple levels of parallelism in a single portable program. With the help of APR's parallelization software, existing Fortran programs can run on clustered SMP systems.

6.3.2 Other development environments for DMP using workstation cluster

The Portland Group supplies SuperCompilers and advanced development tools for high performance computing. With pghpf, existing Fortran code with High Performance Fortran (HPF) directives are translated into code which automatically uses multiple processors in shared and distributed memory parallel systems.

6.3.3 Advantages of workstation cluster

- They share some of the advantages discussed under distributed computing with DCE.
- They are economical.
- Existing parallelization preprocessors maintain source code portability. The parallelization directives and code modifications by the tool are performed on a copy of the original source code.
- They are dynamically scalable. The parallel code can work even on a single workstation.
- They can be used as a test bed for parallel program development for MPPs.
- Current versions of SDP Toolkit and HDF library do not have to be redesigned because there is no memory contention in a distributed environment.

6.3.4 Disadvantages of workstation cluster

- Existing parallelization tools currently support only serial I/O. Forge 90/xHPF appears to be poised for parallel I/O.
- Serial I/O can seriously degrade performance if not completely masked by CPU intensive workload.
- If data volumes are large, data movement among distributed CPUs can degrade performance substantially.
- The speed of interconnect can affect performance. However, ATMs can provide very high bandwidth and low latency.
- Developing code for DMPs is more difficult than for SMPs.
- A separate load balancing software should used in managing a cluster.
- Failure and recovery is more complicated in workstation clusters than in DMP in one cabinet (e.g. Cray J916, etc.).
- A run-time executable needs to be prepared in each workstation. The code should be ported to, compiled and linked in each workstation it is intended to be run in a distributed cluster mode.

6.3.5 Workstation cluster Issues

- DMP Tools are advanced for Fortran but not for C family. Only native parallelization preprocessors are available for C language. Third-party parallelization tools (e.g. Forge 90 and xHPF) do not cover C and its family of languages.
- Parallelization tools currently distribute code among heterogeneous architectures belonging to the same family (e.g. with same word length, byte ordering, etc.).

6.3.6 Performance

In a distributed mode with clustered processors, the performance is affected greatly by the speed of interconnect (Ethernet, FDDI, FDDI+GIGAswitch, ATM, etc.). High speed networks can make it seamless.

6.3.7 Portability

The parallel nature of symmetric architectures is transparent to sequential programs, and most programs can be ported from other systems with little or no change to the source code. Both parallel programs and ordinary sequential programs can run simultaneously on these systems. However, for distributed heterogeneous architectures, portability is governed by the individual platforms the application will be run, and also the underlying communication and resource management software.

6.4 Massively Parallel Processors

As discussed earlier, MPP machines consists of numerous processors and fall under the DMP category (although some architectures like Cray T3D allow for the use of limited shared memory). PDPS prototyping on the MPPs at select NASA HPCC sites is ongoing and will continue until CDR. Therefor, the following sections must be considered preliminary. More detailed results are forthcoming.

6.4.1 Parallelization tools for MPPs

There has been considerable technological advancement in the development and availability of automatic parallelization tools for MPP architectures. APR's parallelization tools available for workstation cluster are also available for nCUBE, Intel Delta, Intel Paragon, IBM SP-2 and Cray T3D. This provides capabilities for parallel applications developed on workstation cluster to be migrated to MPPs relatively painlessly.

6.4.1 Advantages of MPPs

- MPPs allow multiple I/O paths into the system. If the data can be partitioned among different I/O channels, then current MPPs can provide an acceptable solution.
- They have large I/O bandwidths and low latencies that surpasses any external interconnects. The ECS tall pole algorithms could potentially be benefited by MPPs.
- MPPs are very scalable.

6.4.2 Disadvantages of MPPs

- MPPs are expensive. However, smaller MPPs are now available at substantially lower prices than a full-fledged high-end MPP with numerous processors.

- The spectrum of applications for MPPs may be limited. If the large number of processors available on MPPs are not utilized effectively by the applications, the purpose is defeated.
- Developing (or porting existing) applications to MPP architectures require a lot of programmer effort if parallelization is pursued independently without the aid of tools. This is done to extract all the performance an MPP has to offer but at the expense of portability. However, development can be easier via one of the distributed computing modes discussed earlier. Third-party parallelization tools (like APR's Forge 90/xHPF) can decrease programmer effort and increase portability from workstation cluster environment to MPPs.
- Not all MPPs have native parallelization tools that are good. The Cray Research Adaptive Fortran (CRAFT) programming model incorporates both the traditional data parallel and message passing programming styles with the newer work sharing capability. Preliminary prototyping efforts with CRAFT showed promising results for automatically parallelizing applications. However, CRAFT is not GUI-based unlike Forge 90, PFA/PCA on SGI Challenge.

6.4.3 MPP Issues

- Only Forge 90/xHPF from APR provides a common parallelization tool to assist software development on workstation cluster for later migration to MPPs. A single third-party vendor introduces substantial risk to the project in the long term.
- Some native parallelization compilers (CRAFT, etc.) are powerful but limit portability of science software to other hardware.

6.4.4 Performance

MPP systems definitely have the potential for satisfying the need for more performance at less price. Performance depends upon how well the problem lends itself to processing on large number of processors. But they are limited by the programming model used, availability of application software, and limitations on parallelization.

6.4.5 Portability

MPP Fortran dialects hide data communication and inter-processor management issues, but the native tools they offer for data layout are machine specific, presenting an obstacle to porting applications from one MPP to another. Perhaps a language like High Performance Fortran (HPF) will go a long way in solving portability issues.

Despite advances in languages and compilers for MPPs, parallel programming will remain more challenging than on an SMP, especially during debugging and performance tuning. There are more tuning parameters in MPPs (such as data placement, granularity, etc.) if one wishes to completely take advantage of the architecture, albeit at the expense of portability.

As mentioned earlier, APR parallelization tools are available on select MPP hardware that would allow creation of parallel programs in a workstation environment for later migration to MPPs.

7. Recommendations and Conclusions

Prototyping on the distributed/parallel test bed at the ECS STL made it possible to evaluate hands-on the emerging technologies in distributed and parallel computing. The following succinctly represent recommendations and conclusions:

- Both SMP and DMP workstation cluster technologies are viable alternatives to serial computing. However, the shared memory model makes it easier to program on an SMP.
- Parallelization tools for SMP, DMP and more recently select MPPs are very sophisticated. They can guide even novice parallel programmers to set up parallel applications quickly.
- A parallelization strategy should be done at design and implemented when coding. A parallel program will run on a single processor also. Results from prototyping indicate that a parallel program runs more efficiently than a serial program even on a single processor.
- I/O can seriously degrade performance. CPU intensive workload should mask I/O workload to maximize performance. Parallel I/O can elegantly solve I/O bottlenecks. However, no standards have emerged yet.
- The SDP Toolkit and HDF-EOS libraries need modifications to effectively use them in a parallel environment.
- Parallelization tools to program on SMP clusters are available.
- MPPs provide acceptable solutions for very I/O and CPU intensive applications. However, effective utilization of processors are strongly dependent on the application.
- The algorithm must be characterized and carefully matched to the processing paradigm after considering a variety of factors including performance, scalability, portability, maintainability, etc.

8. List of Acronyms

APR	Applied Parallel Research, Inc.
ATM	Asynchronous Transfer Mode
AVHRR	Advanced Very High Resolution Radiometer
CDR	Critical Design Review
CPU	Central Processing Unit
DCE	Distributed Computing Environment
DMP	Distributed Memory Processing
DPS	Data Processing System
ECS	EOSDIS Core System
EOSDIS	Earth Observing System Data and Information System
FDDI	Fiber Distributed Data Interface
GAC	Global Area Coverage
GSFC	Goddard Space Flight Center
GUI	Graphical User Interface
HDF-EOS	Hierarchical Data Format with EOS extensions
HPCC	High Performance Computing Center
HPF	High Performance Fortran
I/O	Input/Output
MIMD	Multiple-Instruction, Multiple-Data
MISD	Multiple-Instruction, Single-Data
MP	Message Passing
MPI	Message Passing Interface
MPP	Massively Parallel Processing
MSFC	Marshall Space Flight Center
MSFC	Marshall Space Flight Center
NASA	National Aeronautics and Space Administration
OO	Object Oriented

OSF	Open Software Foundation
PCA	Power C Analyzer
PDPS	Planning and Processing System
PDR	Preliminary Design Review
PFA	Power Fortran Accelerator
PVM	Parallel Virtual Machine
RPC	Remote Procedure Call
SDP	Science Data Processing
SGI	Silicon Graphics, Inc.
SIMD	Single-Instruction, Single-Data
SISD	Single-Instruction, Single-Data
SM	Shared Memory
SMP	Symmetric Multiprocessing
SPMD	Single Program, Multiple Data
SSM/I	Special Sensor Microwave/Imager
STL	Science and Technology Laboratory